# BUNDESREPUBLIK DEUTSCHLAND





## Prioritätsbescheinigung über die Einreichung einer Patentanmeldung

| | |
|---|---|
| **Aktenzeichen:** | 100 39 277.6 |
| **Anmeldetag:** | 11. August 2000 |
| **Anmelder/Inhaber:** | GfS Systemtechnik GmbH & Co KG, Aachen/DE |
| **Bezeichnung:** | Verfahren für die termingerechte Ausführung einer Zielfunktion |
| **IPC:** | G 06 F 1/14 |

**Die angehefteten Stücke sind eine richtige und genaue Wiedergabe der ursprünglichen Unterlagen dieser Patentanmeldung.**

München, den 28. September 2000
**Deutsches Patent- und Markenamt**
**Der Präsident**
Im Auftrag

Dzierzon

Beschreibung:

**Verfahren für die termingerechte Ausführung einer Zielfunktion**

Die Erfindung betrifft ein Verfahren für die termingerechte Ausführung von Programmschritten durch den Prozessor eines Computers zu vorbestimmten Zeitpunkten, bei dem wiederholt ein Register des Computers ausgelesen und dessen Wert mit einem den vorbestimmten Zeitpunkt repräsentierenden Referenzwert verglichen wird, wobei bei Übereinstimmung des ausgelesenen Werts mit dem Referenzwert die genannten Programmschritte auf dem Prozessor ausgeführt werden. Die Abfolge der termingerecht auszuführenden Programmschritte wird nachfolgend als Zielfunktion bezeichnet.

Im genannten Verfahren wird eine Technik verwendet, die unter der englischen Bezeichnung "Polling" bekannt ist. Beim Polling wird kontinuierlich der Wert eines Registers eines Computers mit einem vorgegebenen Wert verglichen und bei Erreichen des vorgegebenen Werts eine bestimmte Zielfunktion aufgerufen. Bei Computer-Registern, die durch einen Quarz mit einer gleichmäßigen Taktrate inkrementiert bzw. dekrementiert werden, entspricht ihr Zahlenwert immer einem konkreten Zeitwert. Solche Register werden als Zählregister bezeichnet. Somit kann durch Vorgabe des Zahlenwerts des Zählregisters exakt der Zeitpunkt der Funktionsausführung vorgegeben werden. Insbesondere bei der Verwendung des Zählregisters, das mit einer hohen Taktrate inkrementiert bzw. dekrementiert wird, kann der Zeitpunkt mit einer sehr hohen Präzision angegeben werden. Die höchste Taktrate innerhalb eines Computers ist in der Regel der Prozessortakt. Bei modernen Prozessoren wird pro Taktzyklus des Prozessors (also im Prozessortakt) ein Zählregister inkrementiert. Dieses Prozessortakt-Zählregister (kurz: PZR) kann von einem Softwareprogramm ausgelesen werden und somit mit hoher Auflösung die aktuelle Zeit angeben. Ist der Prozessortakt bekannt (z.B. 100MHz), so kann aus jeder PZR-Differenz (z.B. 100000) eine zeitliche Differenz berechnet werden (z.B. 1ms).

Der Nachteil des Polling-Verfahrens liegt darin, daß es nicht multitaskingfähig ist, das heißt, daß es ohne Einbußen an zeitlicher Präzision nicht simultan mit einem oder mehreren anderen Programmen auf dem gleichen Prozessor ausgeführt werden kann. Reines Polling mit höchster Auflösung

erfordert die gesamte Rechenleistung des Prozessors und läßt keine Rechenzeit für andere Programme übrig.

Aufgabe der Erfindung ist es, das eingangs genannte Verfahren multitaskingfähig zu machen.

Diese Aufgabe wird erfindungsgemäß dadurch gelöst, daß das Auslesen des Zählregisters innerhalb einer Startfunktion erfolgt, welche durch den Prozessor als Interrupt–Service-Routine ausgeführt wird, wobei vorzugsweise das Interrupt-Signal mit einem zeitlichem Vorlauf vor dem vorbestimmten Zeitpunkt ausgelöst wird und wobei der zeitliche Vorlauf derart festgelegt wird, daß er größer als der zu erwartende maximale Zeitverzug zwischen dem Anliegen des Interrupt-Signals an dem Interrupt-Eingang des Prozessors und dem Interrupt-Aufruf, d.h. dem Ausführen der Startfunktion, ist.

Das erfindungsgemäße Verfahren kombiniert das eingangs beschriebene Polling mit einem zweiten bekannten Verfahren für den zeitdiskreten Aufruf einer bestimmten Funktion, nämlich dem Interrupt-Aufruf (Interrupt Request). Ein Interrupt-Aufruf ermöglicht allerdings - anders als das beschriebene Polling - eine geringere Präzision bei der Einhaltung des vorbestimmten Termins. Ein Prozessor verfügt in der Regel über verschiedene Interrupt-Eingänge, die unterschiedliche Prioritäten haben. Je nachdem, welche Priorität der verwendete Interrupt-Eingang aufweist, ergibt sich ein kürzerer oder längerer Zeitverzug zwischen dem Anliegen des Interrupt-Signals und der tatsächlichen Ausführung der dem entsprechenden Interrupt-Eingang zugeordneten Folge von Programmschritten, der sogenannten Interrupt-Service-Routine (ISR). Weitere Faktoren, die den Zeitverzug zwischen dem Anliegen des Interrupt-Signals und der Ausführung der ISR beeinflussen, sind die aktuelle Auslastung des Prozessors durch andere auf dem Prozessor ablaufende Tasks, die temporär die Interruptverarbeitung des Prozessors deaktivieren. Schließlich werden Interrupt-Aufrufe mit geringerer zeitlicher Auflösung ausgeführt. Sogenannte Timer-Interrupts, bei denen ein Zeitgeber des Computers das Interrupt-Signal an den Interrupt-Eingang des Prozessors leitet, weisen beispielsweise eine Frequenz auf, die um mehr als Faktor 100 niedriger sein muß als der Prozessortakt, da eine sinnvolle ISR mindestens 100 Prozessorbefehle umfaßt und daher mindestens 100 Prozessortakte zur Ausführung benötigt.

Da ein Interrupt-Aufruf alle auf dem Prozessor ablaufenden Programme nur kurz für die Ausführung der Interrupt-Service-Routine unterbricht, ist er sehr gut geeignet, auf einem Multitasking-System zeitdiskrete Funktionsaufrufe durchzuführen. Die Genauigkeit des Pollings läßt sich mit einem Interrupt-Aufruf indes nicht erreichen. Aus diesem Grund wird gemäß der Erfindung ein Interrupt-Aufruf mit einem herkömmlichen Pollingverfahren kombiniert. Während der Laufzeit des Computers wird ein Wert für den maximalen Zeitverzug zwischen dem Anliegen des Interrupt-Signals an dem Interrupt-Eingang des Prozessors und der Ausführung der Startfunktion ermittelt oder geschätzt. Zumindest um diesen maximalen Zeitverzug vor dem vorbestimmten Zeitpunkt wird durch einen Interrupt-Aufruf das wiederholte Auslesen des Zählregisters und Vergleichen des ausgelesenen Wertes mit dem Referenzwert (Pollingverfahren) durchgeführt.

Durch das Interrupt-Signal, das so frühzeitig ausgelöst wird, daß der Interrupt-Aufruf der Startfunktion auf alle Fälle vor dem vorbestimmten Zeitpunkt ausgeführt wird, ist sichergestellt, daß das Pollingverfahren bei Erreichen des vorbestimmten Zeitpunkts auf dem Prozessor abläuft und die gewünschten Programmschritte der Zielfunktion abgearbeitet werden. Durch den Aufruf der Zielfunktion wird das Polling beendet und der Prozessor kann sich nach dem Abarbeiten der Zielfunktion anderen Aufgaben widmen, bis innerhalb eines erneuten Interrupt-Aufrufes die Startfunktion erneut ausgeführt wird, um zu einem neuen vorbestimmten Zeitpunkt die Zielfunktion aufzurufen.

Damit während des Pollings keine Unterbrechung erfolgt, muß sichergestellt sein, daß während der Laufzeit der Startfunktion die Interrupt-Verarbeitung des Prozessors ausgeschaltet ist.

Wie bereits erwähnt, wird für ein Polling mit optimaler Zeitauflösung das Prozessortakt-Zählregister des zentralen Prozessors (CPU) verwendet. Das Interrupt-Signal wird vorzugsweise durch einen Timer-Interrupt bewirkt, der von einem quarzgetakteten Zeitgeber des Computers ausgelöst wird.

Da der zu erwartende maximale Zeitverzug zwischen dem Anliegen des Interrupt-Signals an dem Interrupt-Eingang und dem hierdurch bewirkten

Interrupt-Aufruf der Startfunktion variiert, ist es sinnvoll, diesen maximalen Zeitverzug kontinuierlich während der Laufzeit des Computers zu ermitteln. Durch Annahme eines zu großen Zeitverzuges würde im statistischen Mittel eine unnötig lange Zeit mit dem Pollingverfahren verbracht.

Der zeitabhängige Wert für den zu erwartenden maximalen Zeitverzug wird vorzugsweise auf der Grundlage des tatsächlichen Zeitverzugs ermittelt. Der tatsächliche Zeitverzug kann durch Auslesen des Zählregisters für den Prozessortakt zu Beginn der Startfunktion ermittelt werden, indem derjenige Wert (in der Einheit des Zählregisters) subtrahiert wird, der dem Zeitpunkt des Interrupt-Signals entspricht. Dieser gemessene tatsächliche Zeitverzug sollte mit einem Sicherheitsfaktor, der zwischen 1,2 und 2 liegt, multipliziert werden. Allerdings sollte ein oberer Grenzwert für den maximalen Zeitverzug und damit für den zeitlichen Vorlauf des Interrupt-Signals festgelegt werden. Wenn von einem zu großen Zeitverzug zwischen dem Interrupt-Signal und dem Abarbeiten der Startfunktion ausgegangen wird, erfolgt der Interrupt-Aufruf durch das erfindungsgemäße Verfahren regelmäßig zu frühzeitig. Hieraus resultiert eine lange Laufzeit des Pollingverfahrens, bevor die Zielfunktion ausgeführt wird. Es verbleibt wenig Rechenzeit für andere Aufgaben des Multitasking-Systems. Dies kann zu einer Lähmung des Betriebssystems führen. Der Anfangswert für den zeitlichen Vorlauf sollte innerhalb eines Testlaufes ermittelt werden, um die Wahrscheinlichkeit eines zu späten Aufrufes der Zielfunktion so gering wie möglich zu halten. Innerhalb des Testlaufes erfolgt kein Aufruf der Zielfunktion, es wird lediglich der tatsächliche Zeitverzug ermittelt um dessen Maximalwert zu bestimmen. Während des Testlaufes sollten 10-100 Aufrufe der Startfunktion durchgeführt werden.

Da bei modernen Computern, insbesondere Personalcomputern, mit einer Vielzahl interner und externer Peripheriegeräte die Interrupt-Eingänge weitgehend für bestimmte Funktionen reserviert sind, ist es unwahrscheinlich, daß für die Durchführung des erfindungsgemäßen Verfahrens ein eigener separater Interrupt-Eingang zur Verfügung steht. Außerdem ist es erforderlich, den Interrupt-Aufruf zu diskreten Zeitpunkten erfolgen zu lassen. Aus diesem Grund bietet es sich an, einen Timer-Interrupt zu verwenden, das heißt ein Interrupt-Signal, das von einem Zeitgeber des Computers ausgelöst wird. Die Timer-Interrupts eines Personalcomputers

werden in der Regel durch sein Betriebssystem für verschiedene Funktionen benutzt. Dies gilt insbesondere für den Timer-Interrupt mit der höchsten Priorität (IRQ 0), der bei IBM-kompatiblen PCs der x86er Reihe u.a. die interne Uhr und die Prozessor-Zeitaufteilung für die verschiedenen Programme (Task Scheduler) steuert. Die Verwendung des höchstpriorisierten Timer-Interrupts hat den Vorteil, daß der maximale Zeitverzug den geringsten Wert annimmt.

Deswegen wird vorzugsweise ein Timer-Interrupt für den Aufruf der Startfunktion verwendet, der von mindestens einem anderen, gleichzeitig auf dem Computer ablaufenden Programm, insbesondere von dem Betriebssystem, mitgenutzt wird. Das Betriebssystem erwartet bei der Ausführung des genannten Timer-Interrupts den Ablauf einer bestimmten eigenen Interrupt-Service-Routine. Diese, dem mitgenutzten Timer-Interrupt zugeordnete Service-Routine wird nachfolgend "Originalfunktion" genannt. Um die Startfunktion anstelle der durch den Timer-Interrupt ursprünglich aufgerufenen Originalfunktion ablaufen zu lassen, wird aus der Interrupt-Tabelle, welche die Adressen der den verschiedenen Interrupt-Eingängen zugeordneten Service-Routinen enthält, die Adresse der Originalfunktion ausgelesen und durch die Adresse der Startfunktion ersetzt.

Die Auslöserate des IRQ 0 kann bei modernen Multitasking-Betriebssystemen durch Umprogrammieren des zugehörigen Timers verändert werden. Ein unerwartetes Umprogrammieren des Timers würde eine empfindliche Störung des erfindungsgemäßen Verfahrens bedeuten, da hierdurch die Intervall-Länge zwischen zwei Interrupt-Aufrufen geändert wird. Möglicherweise würde hierdurch ein Interrupt-Aufruf für den Beginn des Pollingverfahrens erst nach dem Zeitpunkt erfolgen, zu dem die Zielfunktion ausgeführt werden sollte. Aus diesem Grund ist es sinnvoll, ein Verstellen der Taktrate des verwendeten Timer-Interrupts auszuschließen. Dies kann dadurch erfolgen, daß dem Betriebssystem der Bedarf der maximalen Interrupt-Taktrate mitgeteilt wird. Für den Timer-Interrupt IRQ 0 bietet die API der Betriebssysteme dazu in der Regel entsprechende Funktionen an (z.B. Multimedia-Funktionen). Beim Aufrufen der Startfunktion und während des Ablaufs des erfindungsgemäßen Verfahrens muß die eingestellte Taktrate nicht beibehalten werden. Die Taktrate kann durch die das Verfahren durchführenden Programmodule beliebig verstellt werden, wenn sie vor

dem Ende des Verfahrens wieder auf die eingestellte maximale Taktrate zurückgestellt wird.

Durch das Festlegen der maximalen Taktrate für den Timer-Interrupt ist es erforderlich, bei jedem Aufruf dieses Timer-Interrupts die Interrupt-Service-Routine (Originalfunktion) auszuführen, von deren Ausführung das Betriebssystem ausgeht. Das erfindungsgemäße Verfahren arbeitet optimal, wenn der Timer umprogrammiert wird. Beim Umprogrammieren muß jedoch darauf geachtet werden, daß die Originalfunktion zu den Zeitpunkten aufgerufen wird, die das Betriebssystem seinerseits durch die Programmierung des Timers festgelegt hat. Daher muß das erfindungsgemäße Computerprogramm eine Liste mit den vorbestimmten Zeitpunkten für die Ausführung der Zielfunktion und eine Liste mit den Zeitpunkten für die Ausführung der Originalfunktion erstellen. Das erfindungsgemäße Verfahren arbeitet beide Listen gleichzeitig ab. Das Polling und die Vorgabe des Zeitpunktes des Interrupt-Signals durch geeignete Programmierung des Timers kann dabei sowohl zum Aufruf der Zielfunktion als auch zum Aufruf der Originalfunktion verwendet werden. Beim Eintritt eines vorbestimmten Zeitpunktes wird anhand der Liste, durch die der entsprechende Zeitpunkt vorgegeben ist, entschieden, ob die Zielfunktion oder die Originalfunktion aktiviert werden soll. Sind auf beiden Listen identische Zeitpunkte vorhanden, so werden beim Eintritt eines solchen Zeitpunktes die Zielfunktion und die Originalfunktion unmittelbar nacheinander aktiviert.

Das Betriebssystem erwartet am Ende der durch den Timer-Interrupt IRQ 0 ausgeführten Originalfunktion einen bestimmten Registerinhalt der Prozessorregister. Aus diesem Grund ist es erforderlich, die Registerinhalte der Prozessorregister zu Beginn der betriebssystem-fremden Startfunktion abzuspeichern und am Ende der Startfunktion wieder zurück in die Prozessorregister zu schreiben. Durch PUSH-Befehle werden die Registerinhalte zu Beginn der Startfunktion auf den Stapel-Speicher (Stack) des Computers geschrieben. Das Rücklesen in die Prozessorregister erfolgt durch POP-Befehle.

Der Interrupt-Controller überwacht jeweils die aktuell ausgeführten Interrupts. Mit Hilfe des Interrupt-Controllers läßt sich ermitteln, ob der entsprechende Interrupt-Aufruf IRQ 0 durch den Timer erfolgte

(Hardware-Interrupt) oder durch das Betriebssystem (Software-Interrupt), welches gelegentlich diesen Interrupt-Aufruf aus der Interrupt-Service-Routine selbst durchführt. Die Unterscheidung zwischen Hardware- und Software-Interrupt läßt sich dadurch bewirken, daß zu Beginn der Startfunktion durch Abfrage eines Registers des Interrupt-Controllers der aktuell ausgeführte Interrupt ermittelt wird und anschließend die Bearbeitung des aktuellen Interrupt-Aufrufs durch einen End-of-Interrupt-Befehl (EOI) bestätigt wird. Stellt die Startfunktion anhand des Registerinhalts des Interrupt-Controllers fest, daß aktuell der IRQ 0 ausgeführt wird, so wird erkannt, daß es sich um einen Hardware-Interrupt handelt. Wird die Bearbeitung eines anderen Interrupt-Aufrufs mitgeteilt, weil durch den EOI-Befehl der Startfunktion die Bearbeitung des IRQ 0 bereits als beendet gilt, ist erkennbar, daß es sich um einen Software-Interrupt handelt. Die Startfunktion wird in diesem Fall die von dem Betriebssystem erwarteten Programmschritte aktivieren.

Wogegen die Zielfunktion durch einen sogenannten Funktionsaufruf (engl.: CALL) aktiviert wird, wobei nach Ablauf aller Programmschritte eine Rückkehr zur Startfunktion erfolgt, wird die Originalfunktion durch einen Sprungbefehl (engl.: JUMP) aktiviert, so daß am Ende des Ablaufs der Originalfunktion keine Rückkehr zur Sprungstelle erfolgt. So ist sichergestellt, daß das Betriebssystem am Ende des Ablaufs der Originalfunktion keinen Hinweis darauf erkennt, daß die Originalfunktion nicht unmittelbar durch den Interrupt-Aufruf, sondern über die Startfunktion aktiviert wurde.

Um eine Unterbrechung der Startfunktion und der durch diese aufgerufenen Funktionen zu vermeiden, muß sichergestellt sein, daß während der Laufzeit der Startfunktion die Interrupt-Verarbeitung des Prozessors deaktiviert ist. Hierdurch wird vermieden, daß durch eine Unterbrechung der Startfunktion an einer ungünstigen Stelle das Betriebssystem durch den Ablauf der Startfunktion gestört wird. Schließlich hat die Startfunktion die Aufgabe, Laufzeitkonflikte während ihres eigenen Ablaufs zu ermitteln. Beispielsweise kann während des Ablaufs der Startfunktion bereits der nächste Timer-Interrupt anliegen. Auch kann die Laufzeit der durch die Startfunktion aktivierten Funktionen (Zielfunktion und/oder Originalfunktion) so groß sein, daß am Ende der Funktionen der nächste vorbestimmte Zeitpunkt für die Durchführung der Zielfunktion bereits verstrichen ist. Schließlich kann

zwischen den Interrupt-Aufrufen des erfindungsgemäßen Verfahrens zu wenig Zeit verbleiben, damit das Betriebssystem und die dadurch ausgeführten Programme stabil ablaufen. Alle diese Fehler können von der Startfunktion überprüft und an die Zielfunktion mitgeteilt werden. Je nach Anwendungsfall, das heißt, je nach den Programmschritten der Zielfunktion sowie deren Wichtigkeit, kann auf die entsprechenden Fehlermeldungen reagiert werden. Beispielsweise kann eine Ausführung der Zielfunktion ausgelassen werden und diese Tatsache in einem Fehlerprotokoll registriert werden, wenn der Ablauf der Zielfunktion keine übergeordnete Bedeutung hat. Ist der Ablauf der Zielfunktion zu einem bestimmten Zeitraum von überragender Wichtigkeit, können alle anderen Programmschritte des Computers bis auf weiteres unterbrochen werden. Während dieses Zeitraums können das Betriebssystem des Computers und die hierdurch ausgeführten Programme äußerst langsam werden und temporär zum Stillstand kommen.

Nachfolgend werden Ausführungsbeispiele der Erfindung unter Bezugnahme auf die beigefügten Zeichnungen erläutert. Die Zeichnungen zeigen in

Fig. 1 ein Ablaufdiagramm für den Ablauf eines Ausführungsbeispiels des erfindungsgemäßen Verfahrens, wenn der Timer nicht mitgenutzt wird und frei programmierbar ist,

Fig. 2 ein Schema für den Ablauf eines Ausführungsbeispiels des erfindungsgemäßen Verfahrens unter Mitnutzung eines von einem anderen Programm genutzten Timers,

Fig. 3 ein Ablaufdiagramm für den Ablauf nach Fig. 2, wenn der mitgenutzte Timer frei programmierbar ist und

Fig. 4 ein Ablaufdiagramm für den Ablauf nach Fig. 2, wenn der mitgenutze Timer eine feste Taktrate aufweist.

Die Fig. 1 zeigt den prinzipiellen Ablauf eines Programms, welches von dem erfindungsgemäßen Verfahren Gebrauch macht. Durch einen Interrupt-Aufruf wird als Interrupt-Service-Routine die genannte Startfunktion aufgerufen. Diese liest das Prozessortakt-Zählregister (PZR) aus und vergleicht den dadurch ermittelten Zeitpunkt mit dem Zeitpunkt des Interrupt-Signals. Die Zeitdifferenz stellt den aktuellen Zeitverzug dar. Ist der aktuelle Zeitverzug größer als der eingestellte Wert des zeitlichen Vorlaufs des Interrupt-Signals, so muß der Wert für den zeitlichen Vorlauf zumindest

auf den gemessenen aktuellen Zeitverzug eingestellt werden. Vorzugsweise sollte ein Sicherheitsfaktor von 1,2 bis 2 berücksichtigt werden, so daß der zeitliche Vorlauf größer ist als der gemessene aktuelle Zeitverzug.

Anschließend durchläuft das Programm ein Pollingverfahren, in dem kontinuierlich das PZR ausgelesen und der gelesene Wert mit dem Wert verglichen wird, der den vorbestimmten Zeitpunkt für den Ablauf der Zielfunktion repräsentiert. Bei Übereinstimmung wird die Zielfunktion aufgerufen. Im Anschluß wird der Timer auf den nächsten vorbestimmten Zeitpunkt abzüglich des zeitlichen Vorlaufs programmiert, so daß wieder rechtzeitig vor dem nächsten vorbestimmten Zeitpunkt mittels der Startfunktion das Pollingverfahren durchgeführt wird.

Wie in Fig. 2 erkennbar, kann zur Durchführung des erfindungsgemäßen Verfahrens ein Timer verwendet werden, der eigentlich ausschließlich für die Durchführung einer bestimmten Interrupt-Service-Routine (Originalfunktion) des Betriebssystems vorgesehen ist. Die exklusive Nutzung durch das Betriebssystem ist in der linken Hälfte der Fig. 2 dargestellt. Bei dem Ablauf des erfindungsgemäßen Verfahrens wird die Startfunktion als durch den Timer-Interrupt aktivierte Interrupt-Service-Routine zwischengefügt. Hierzu wird aus der Interrupt-Tabelle im Speicher des Computers die Adresse der Originalfunktion ausgelesen und mit der Adresse der Startfunktion überschrieben. Diese führt alternativ oder - wie nachfolgend erläutert - seriell die Zielfunktion und/oder die Originalfunktion aus.

Die Fig. 3 zeigt den Ablauf des erfindungsgemäßen Verfahrens bei mitgenutztem Timer für den Fall, daß der Timer durch die Startfunktion umprogrammierbar ist. In diesem Fall ermittelt die Startfunktion , ob zum eingetretenen Zeitpunktdie Originalfunktion aktiviert (kein Zieltermin) oder die Zielfunktion aufgerufen (Zieltermin) werden soll. Das Programm, welches das erfindungsgemäße Verfahren durchführt, stellt hierzu eine Liste mit den vorbestimmten Zeitpunkten zur Aktivierung der Originafunktion und eine Liste mit den vorbestimmten Zeitpunkten zum Aufrufen der Zielfunktion zu Verfügung, anhand derer die Startfunktion die erforderliche Unterscheidung treffen kann. Die Zielfunktion wird mit einem "CALL"-Befehl aufgerufen. Am Ende der Zielfunktion erfolgt ein Return (Rücksprung) zur Startfunkti-

on, welche durch einen Interrupt-Return (IRET) an die durch den Interrupt unterbrochene Programmstelle zurückkehrt. Die Originalfunktion wird von der Startfunktion mittels eines JUMP-Befehls aktiviert, nachdem in oben beschriebener Weise der Inhalt der Prozessorregister gesichert wurde. Am Ende der Originalfunktion erfolgt automatisch ein IRET-Befehl.

Die Fig. 4 zeigt den praktischen Fall, daß ein mit einer festen Taktrate getakteter Timer, der von dem Betriebssystem mitgenutzt wird, den Interrupt-Aufruf durchführt. Das Betriebssystem geht in diesem Fall davon aus, daß auf jeden Interrupt-Aufruf der Ablauf der Originalfunktion folgt. Andernfalls würde die Zeitbasis des Betriebssystems nicht korrekt arbeiten, was zum zu schnellen oder zu langsamen Weiterzählen der Systemuhr führt. Das Programm, welches das erfindungsgemäße Verfahren durchführt, stellt der Startfunktion wiederum eine Liste der jeweiligen vorbestimmten Zeitpunkte, zu denen die Zielfunktion aufgerufen werden soll, zur Verfügung. Die Startfunktion liest zunächst das Prozessortakt-Zählregister aus und ermittelt, ob in einem zeitlichen Abstand, der zumindest dem zu erwartenden maximalen Zeitverzug entspricht, vor dem nächsten vorbestimmten Zeitpunkt ein erneuter Interrupt-Aufruf (IRQ) erfolgt. Ist dies der Fall, so kann durch die Startfunktion der Sprung zur Originalfunktion ausgeführt werden, welche anschließend zu dem unterbrochenen Programmablauf zurückkehrt. Erfolgt kein Interrupt-Aufruf vor dem nächsten vorbestimmten Zeitpunkt, ruft die Startfunktion die Polling-Schleife auf, welche aus dem Auslesen des Prozessortakt-Zählregisters und dem Vergleichen des gelesenen Wertes mit dem Wert für den vorbestimmten Zeitpunkt besteht. Bei Übereinstimmung der beiden Werte wird die Zielfunktion aufgerufen und der nächste vorbestimmte Zeitpunkt in die Bearbeitung genommen. Anschließend wird ermittelt, ob ein Interrupt-Aufruf vor dem nun aktuellen vorbestimmten Zeitpunkt erfolgt. Ist dies nicht der Fall, wird erneut die Polling-Schleife durchgeführt, bis die Zielfunktion aufgerufen wird. Auf diese Weise kann infolge eines Interrupt-Aufrufs mehrfach das Polling durchgeführt werden, bevor der Sprung zur Originalfunktion erfolgt.

Abschließend wird die Umsetzung des in Fig. 1-4 dargestellten erfindungsgemäßen Verfahrens auf einem IBM-kompatiblen Personalcomputer (PC) mit einem x86-Prozessor beschrieben. Als Softwareplattform dienen die Betriebssysteme mit 32-bit Technologie der Firma Microsoft (Windows

95/98/Me/NT/2000). Diese Kombination aus Hardware- und Softwareplattform ist weltweit am häufigsten anzutreffen.

Im IBM-kompatiblen PC sind zwei kaskadierte Interrupt-Controller vom Typ 8259A im Einsatz, die 15 IRQs priorisiert auf den INT-Pin des x86 multiplexen. Der höchstpriorisierte IRQ ist der IRQ 0. An den IRQ 0 ist der Ausgang eines Timers vom Typ 8254 angeschlossen. Der IRQ 0 kann deswegen als Timer-Interrupt verwendet werden. Der Timer 8254 kann in 65536 Frequenzstufen von 18,206 Hz bis 1,193182 MHz programmiert werden. Als PZR kann bei den x86 ab dem Pentium (eingetragene Marke der Firma INTEL) der Time Stamp Counter (kurz: TSC) verwendet werden.

Die Windows Betriebssysteme nutzen den Timer 8254 zum Weiterzählen der Uhrzeit und zur Zeitscheibensteuerung des preemptiven Multitasking-Schedulers. Je nach Anforderung der Zeitauflösung von Applikationen wird der Timer 8254 von dem Betriebssystem fortwährend umprogrammiert, jedoch nur bis zu einer maximalen Rate von ca. 1kHz. Diese maximale Rate kann erzwungen werden durch den Aufruf der WIN32-API Funktion „timeBeginPeriod“ aus der Multimediabibliothek. Durch diesen Aufruf wird die Umprogrammierung des Timers 8254 durch das Betriebssystem vermieden.

Durch Überschreiben des Interrupt-Gates in der IDT beim Index 0x50 (Windows 95/98/Me) oder Index 0x30 (Windows NT/2000) wird die Originalfunktion durch die Startfunktion als neue ISR ersetzt. Dabei ist unter Windows 95/98/Me zu beachten, daß mehrere IDTs verwendet werden, zwischen denen fortwährend umgeschaltet wird. Daher sind die Interrupt-Gates in allen IDTs zu überschreiben.

Die Startfunktion als ISR muß vor dem Sprung in die Originalfunktion dafür sorgen, daß sie nicht den Ablauf des Betriebssystems stört. Dazu sind zu Beginn der Startfunktion folgende Registerinhalte auf dem Stack (Stapelspeicher) zu retten: EAX, EBX, ECX, EDX, ESP, EBP, ESI, EDI, EFLAGS, DS, ES, FS. Dann ist über den Assembler-Befehl „cld“ das Direction-Flag zu löschen, um bei Stringoperationen stets die Indexregister zu inkrementieren statt zu dekrementieren. Wegen der DOS-Kompatibilität der Windows Betriebssysteme und der deswegen konstruierten virtuellen

DOS-Umgebungen ist bei Eintritt der ISR lediglich die zur Ausführung der ISR korrekte Auswahl des bei der linearen Adresse Null beginnenden (engl.: Flat) Codesegmentes (CS) gewährleistet. DS, ES und FS können bei Unterbrechung einer virtuellen DOS-Umgebung für die Ausführung der ISR Werte enthalten, die sich nicht auf Flat-Segmente beziehen. Daher sind DS und ES auf ein Flat-Datensegment zu setzen (Windows 95/98/Me 0x30, Windows NT/2000 0x10) und FS entsprechend auf ein Flat-Codesegment (Windows 95/98/Me 0x28, Windows NT/2000 0x08). Damit sind die notwendigen Vorbereitungen zur Ausführung der ISR getroffen. Bevor die Originalfunktion angesprungen werden kann, müssen die zuvor gespeicherten Registerinhalte vom Stack wieder in die Register zurückgeschrieben werden. Hiervon ist auch das Datensegmentregister DS betroffen. Die Adresse der Originalfunktion ist jedoch in einer Datensegmentvariablen abgelegt, deren Zugriff nicht funktioniert, wenn die ISR nicht in einem Flat-Datensegment aufgerufen wurde. Um dieses Problem zu umgehen, muß der Sprung in die Originalfunktion über das Codesegment adressiert werden, denn eine ISR wird stets in einem Flat-Codesegment ausgeführt.

Durch diese Maßnahmen ist sichergestellt, daß die Startfunktion keine den Ablauf der Originalfunktion oder des Betriebssystems beeinträchtigende Spuren hinterläßt.

Die x86-Prozessoren verfügen neben Hardware-Interrupts auch über Software-Interrupts. Diese werden von der Software selbst ausgelöst und führen zum Aufruf der entsprechenden ISR. Die Originalfunktion macht von diesen Software-Interrupts Gebrauch, indem sie sich selbst mehrfach aufruft. Da statt ihrer selbst die Startfunktion in die IDT eingetragen ist, wird die Startfunktion mehrfach aufgerufen, ohne daß ein Timer-Interrupt vorliegt. Die Startfunktion muß unterscheiden können, ob sie von einem Hardware-Interrupt oder von einem Software-Interrupt aufgerufen wurde, da die Software-Interrupts nicht das Polling und die Ausführung der Zielfunktion aktivieren dürfen. Bei einem Software-Interrupt muß die Startfunktion lediglich in die Originalfunktion springen. Ob die Startfunktion von einem Hardware-Interrupt oder einem Software-Interrupt aufgerufen wurde, kann durch Abfrage des Interrupt-Controllers ermittelt werden. Der Interrupt-Controller 8259A verfügt über ein In-Service-Register, das darüber Auskunft gibt, welcher Interrupt gerade behandelt wird. Durch Abfragen des In-

Service-Registers kann die Startfunktion ermitteln, ob es sich um einen Hardware-Interrupt oder einen Software-Interrupt handelt. Das funktioniert jedoch nur, wenn der Hardware-Interrupt über einen End-of-Interrupt (kurz: EOI) Befehl an den Interrupt-Controller 8259A bereits quittiert wurde. Um zwischen Hardware-Interrupts und Software-Interrupts sicher zu unterscheiden, muß die Startfunktion unmittelbar nach der Auswertung des In-Service-Registers im 8259A einen spezifischen EOI an den Interrupt-Controller 8259A leiten, was zum Löschen des entsprechenden Bits im In-Service-Register führt. Genau dieses Bit wird die Startfunktion abfragen, wenn sie durch einen Software-Interrupt von der Originalfunktion aufgerufen wird. Dadurch kann sie sicher zwischen Hardware-Interrupts und Software-Interrupts unterscheiden.

Die Startfunktion sollte - wie oben erwähnt - auch den Interrupt**vor**konflikt (zu Beginn der Startfunktion liegt bereits der nächste Timer-Interrupt an) und den Interrupt**nach**konflikt (am Ende der Startfunktion liegt bereits der nächste Timer-Interrupt an) erkennen. Dazu muß sie ebenfalls direkt auf den Interrupt-Controller 8259A zugreifen. Der Interrupt-Controller 8259A verfügt über ein Interrupt-Request-Register, das darüber Auskunft gibt, welche Interrupts „in der Warteschlange" stehen. Steht zu Beginn der Startfunktion bereits der IRQ 0 wieder in der Warteschlange, so liegt ein Interruptvorkonflikt vor. Entsprechend liegt ein Interruptnachkonflikt vor, wenn am Ende der Startfunktion bereits der IRQ 0 wieder in der Warteschlange steht.

Die Konsequenzen, die aus einem ermittelten Interruptkonflikt gezogen werden, hängen von den Programmschritten der Zielfunktion ab. Die Startfunktion sendet lediglich eine Fehlermeldung an die Zielfunktion, welche dann die erforderlichen Maßnahmen (z.B. Abbrechen der Zielfunktion) ergreift.

* * * * * * *

Ansprüche:

1. Verfahren für die termingerechte Ausführung von Programmschritten (Zielfunktion) durch den Prozessor eines Computers zu vorbestimmten Zeitpunkten, bei dem wiederholt ein Register des Computers ausgelesen und dessen Wert mit einem den vorbestimmten Zeitpunkt repräsentierenden Referenzwert verglichen wird, wobei bei Übereinstimmung des ausgelesenen Werts mit dem Referenzwert die Zielfunktion auf dem Prozessor ausgeführt wird, **dadurch gekennzeichnet**, daß das Auslesen des Registers innerhalb einer Startfunktion durchgeführt wird, welche durch den Prozessor als Interrupt-Service-Routine ausgeführt wird.

2. Verfahren nach Anspruch 1 , **dadurch gekennzeichnet**, daß das Interrupt-Signal mit einem zeitlichen Vorlauf vor dem vorbestimmten Zeitpunkt ausgelöst wird und daß der zeitliche Vorlauf derart festgelegt wird, daß er größer als der zu erwartende maximale Zeitverzug zwischen dem Anliegen des Interrupt-Signals an dem Interrupt-Eingang des Prozessors und der Ausführung der Startfunktion ist.

3. Verfahren nach Anspruch 1 oder 2, **dadurch gekennzeichnet**, daß als Register ein Zählregister verwendet wird.

4. Verfahren nach einem der vorangehenden Ansprüche, **dadurch gekennzeichnet**, daß als Zählregister das Prozessortakt-Zählregister (PZR) des zentralen Prozessors (CPU) des Computers verwendet wird.

5. Verfahren nach einem der vorangehenden Ansprüche, **dadurch gekennzeichnet**, daß das Interrupt-Signal durch einen Zeitgeber des Computers als Timer-Interrupt ausgelöst wird.

6. Verfahren nach einem der vorangehenden Ansprüche, **dadurch gekennzeichnet**, daß der zu erwartende maximale Zeitverzug kontinuierlich während der Laufzeit des Computers ermittelt wird.

7. Verfahren nach Anspruch 6, **dadurch gekennzeichnet**, daß der Wert für den maximalen Zeitverzug auf der Grundlage des tatsächlichen Zeitverzugs ermittelt wird, welcher durch Auslesen des Zählregisters zu Beginn der

Startfunktion und durch Subtraktion des Wertes, der den Zeitpunkt des entsprechenden Interrupt-Signals darstellt, bestimmt wird.

8. Verfahren nach Anspruch 7, **dadurch gekennzeichnet**, daß der zu erwartende maximale Zeitverzug durch Multiplikation des tatsächlichen Zeitverzugs mit einem Sicherheitsfaktor, der beispielsweise zwischen 1,2 und 2 liegt, ermittelt wird.

9. Verfahren nach einem der Ansprüche 6 bis 8, **dadurch gekennzeichnet**, daß beim Überschreiten eines oberen Grenzwertes durch den ermittelten maximalen Zeitverzug eine Fehlermeldung erfolgt.

10. Verfahren nach Anspruch 6 bis 8, **dadurch gekennzeichnet**, daß beim Überschreiten des oberen Grenzwertes durch den ermittelten maximalen Zeitverzug der Wert für den zeitlichen Vorlauf gleich dem Grenzwert gesetzt wird.

11. Verfahren nach einem der Ansprüche 5 bis 10, **dadurch gekennzeichnet**, daß ein Timer-Interrupt verwendet wird, der von anderen, gleichzeitig auf dem Computer ablaufenden Programmen, insbesondere dem Betriebssystem, für den Aufruf einer Originalfunktion verwendet wird.

12. Verfahren nach Anspruch 11, **dadurch gekennzeichnet**, daß aus der Interrupt-Tabelle, welche die Adressen der den verschiedenen Interrupt-Eingängen zugeordneten Service-Routinen enthält, die Adresse der Originalfunktion ausgelesen und durch die Adresse der Startfunktion ersetzt wird.

13. Verfahren nach Anspruch 11 oder 12, **dadurch gekennzeichnet**, daß durch einen Interrupt-Aufruf mittels der Startfunktion sowohl die termingerecht auszuführende Zielfunktion als auch die Originalfunktion ausgeführt werden.

14. Verfahren nach einem der Ansprüche 11 bis 13, **dadurch gekennzeichnet**, daß der Timer durch das Betriebssystem auf verschiedene Taktraten einstellbar ist und vor Beginn des Verfahrens auf die maximale Taktrate eingestellt wird.

15. Verfahren nach Anspruch 14, **dadurch gekennzeichnet**, daß die Taktrate des Timers durch das Verfahren selbst verändert wird und vor dem Ende des Verfahrens wieder auf die maximale Taktrate zurückgestellt wird.

16. Verfahren nach einem der Ansprüche 11 bis 15, **dadurch gekennzeichnet**, daß eine Liste mit den vorbestimmten Zeitpunkten für die Ausführung der Zielfunktion und eine Liste mit den Zeitpunkten, an denen das Interrupt-Signal ausgelöst wird, erstellt wird, daß die Startfunktion den nächsten Zeitpunkt der Ausführung der Zielfunktion mit dem Zeitpunkt des nächsten Interrupt-Signals vergleicht und eine Ausführung der Originalfunktion veranlaßt, wenn das nächste Interrupt-Signal um den maximalen Zeitverzug vor dem Zeitpunkt der Ausführung der Zielfunktion liegt.

17. Verfahren nach einem der vorangehenden Ansprüche, **dadurch gekennzeichnet**, daß zu Beginn der Startfunktion die Registerinhalte der Prozessorregister, die durch die Startfunktion verändert werden, auf den Stapel-Speicher (Stack) des Computers geschrieben und am Ende der Startfunktion wieder in die Register zurückgeschrieben werden.

18. Verfahren nach einem der vorangehenden Ansprüche, **dadurch gekennzeichnet**, daß zu Beginn der Startfunktion durch Abfrage eines Registers des Interrupt-Controllers der aktuell ausgeführte Interrupt ermittelt wird und anschließend die Bearbeitung des aktuellen Interrupt-Aufrufs durch einen End-of-Interrupt-Befehl (EOI) bestätigt wird.

19. Verfahren nach einem der vorangehenden Ansprüche, **dadurch gekennzeichnet**, daß die Originalfunktion durch einen Sprungbefehl (jump) mittels der Startfunktion aktiviert wird.

20. Verfahren nach einem der vorangehenden Ansprüche, **dadurch gekennzeichnet**, daß die Startfunktion während ihrer Laufzeit ermittelt, ob ein weiteres Interrupt-Signal am Interrupt-Eingang anliegt und in diesem Fall eine Fehlermeldung an die Zielfunktion abgibt.

21. Softwareprogramm-Produkt zum Laden in den Arbeitsspeicher eines durch ein Betriebssystem betriebenen Computers mit einem Prozessor und einem Zählregister, **dadurch gekennzeichnet**, daß es eine Programmschritt-

Folge zur Durchführung eines Verfahrens gemäß einem der vorangehenden Ansprüche umfaßt.

22. Maschinenlesbarer Datenträger mit einem auf dem Datenträger abgespeicherten Softwareprogramm-Produkt nach Anspruch 21.

* * * * * * *

Zusammenfassung:

Die Erfindung betrifft ein Verfahren für die termingerechte Ausführung von Programmschritten (Zielfunktion) durch den Prozessor eines Computers zu vorbestimmten Zeitpunkten, bei dem wiederholt ein Register des Computers ausgelesen und dessen Wert mit einem den vorbestimmten Zeitpunkt repräsentierenden Referenzwert verglichen wird, wobei bei Übereinstimmung des ausgelesenen Werts mit dem Referenzwert die Zielfunktion auf dem Prozessor ausgeführt wird.

Im genannten Verfahren wird eine Technik verwendet, die unter der englischen Bezeichnung "Polling" bekannt ist.

Der Nachteil des Polling-Verfahrens liegt darin, daß es nicht multitaskingfähig ist. Dieser Nachteil soll durch die vorliegende Erfindung beseitigt werden.

Diese Aufgabe wird dadurch gelöst, daß das Auslesen des Registers innerhalb einer Startfunktion durchgeführt wird, welche durch den Prozessor als Interrupt–Service-Routine ausgeführt wird.

FIG. 1

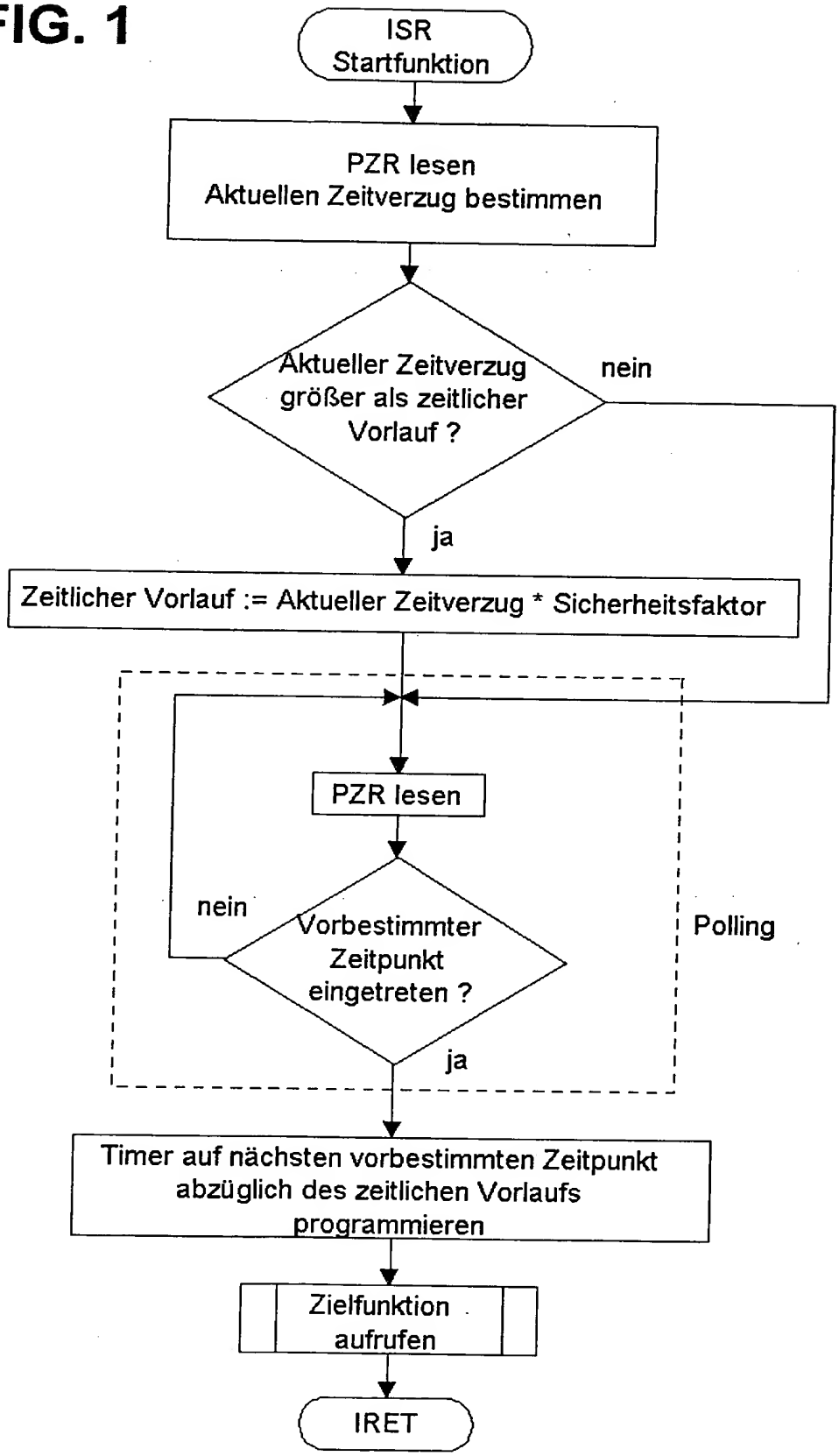

ISR
Startfunktion
PZR lesen
Aktuellen Zeitverzug bestimmen
Aktueller Zeitverzug
größer als zeitlicher
Vorlauf ?
nein
ja
Zeitlicher Vorlauf := Aktueller Zeitverzug * Sicherheitsfaktor
PZR lesen
nein
Vorbestimmter
Zeitpunkt
eingetreten ?
ja
Polling
Timer auf nächsten vorbestimmten Zeitpunkt
abzüglich des zeitlichen Vorlaufs
programmieren
Zielfunktion
aufrufen
IRET

FIG. 2

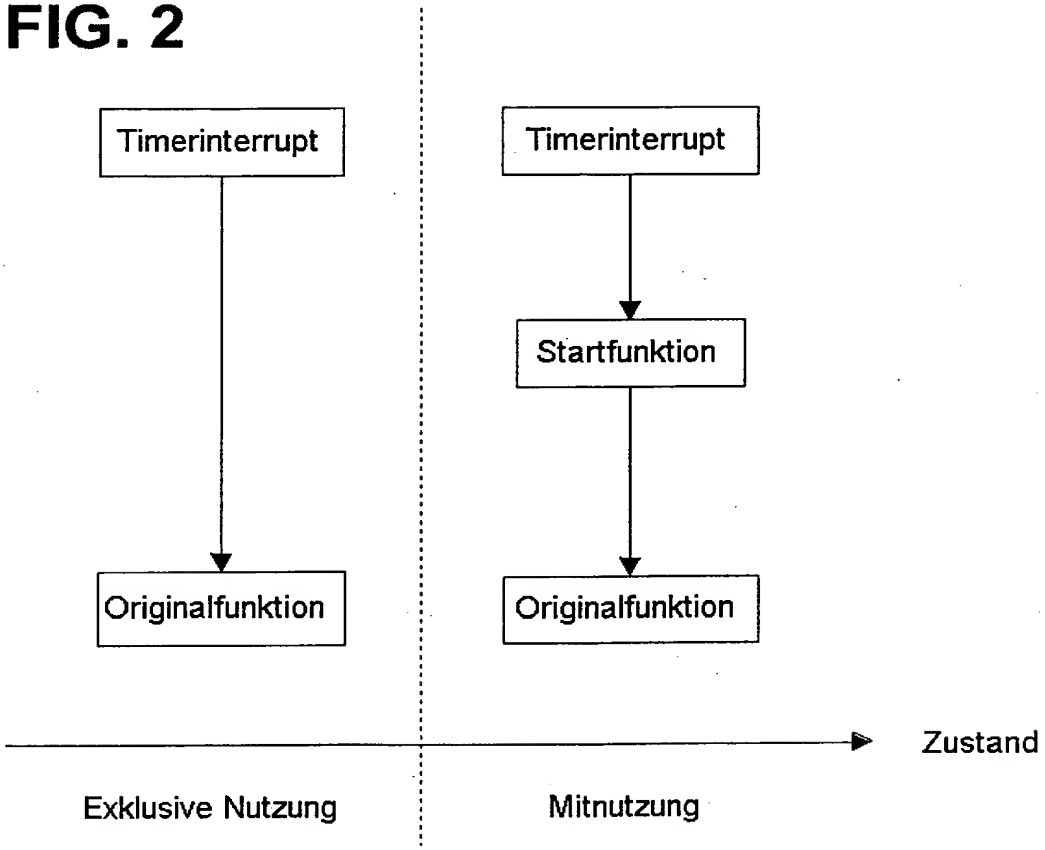

Timerinterrupt
Timerinterrupt
Startfunktion
Originalfunktion
Originalfunktion
Zustand
Exklusive Nutzung
Mitnutzung

FIG. 3

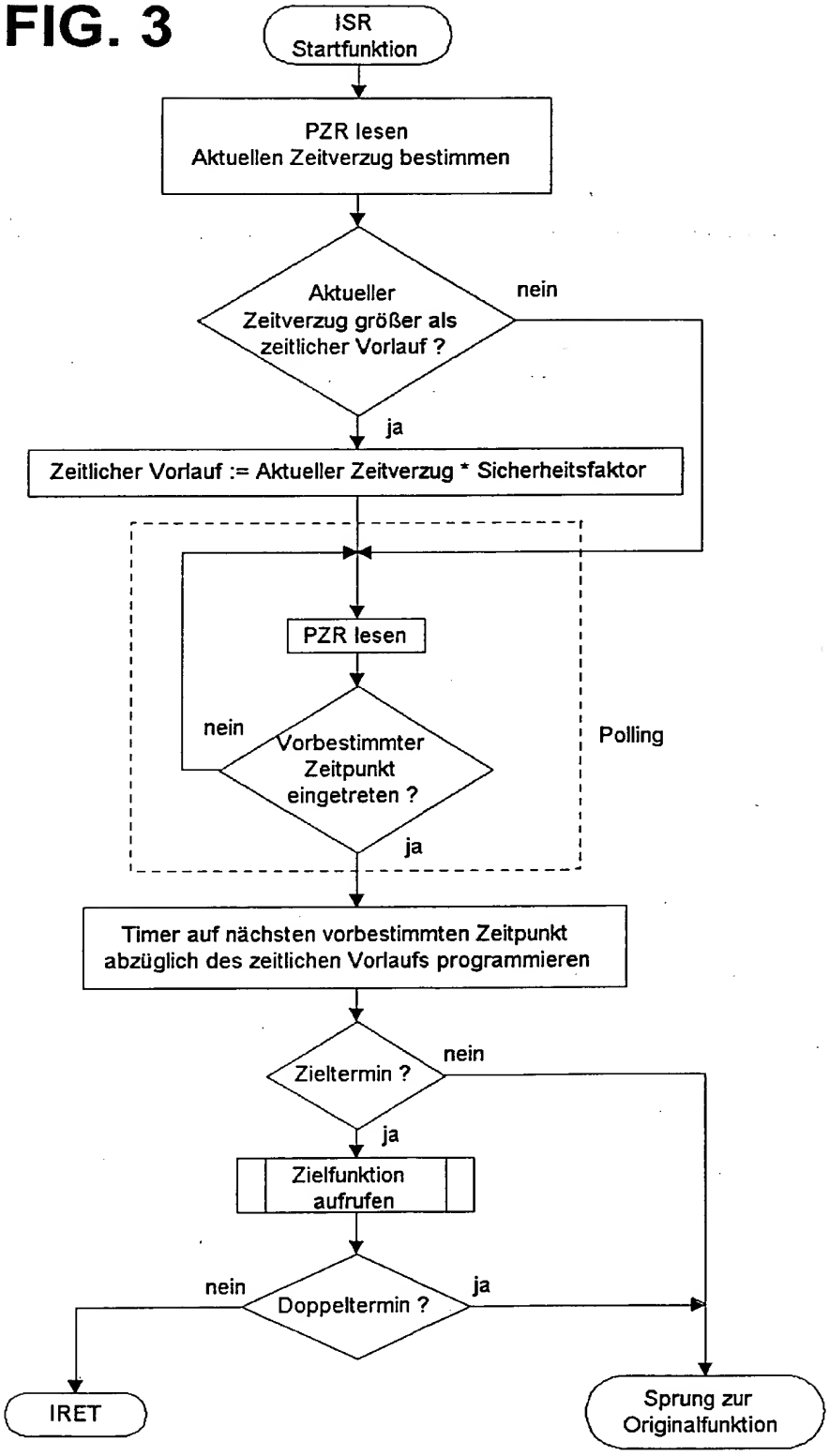
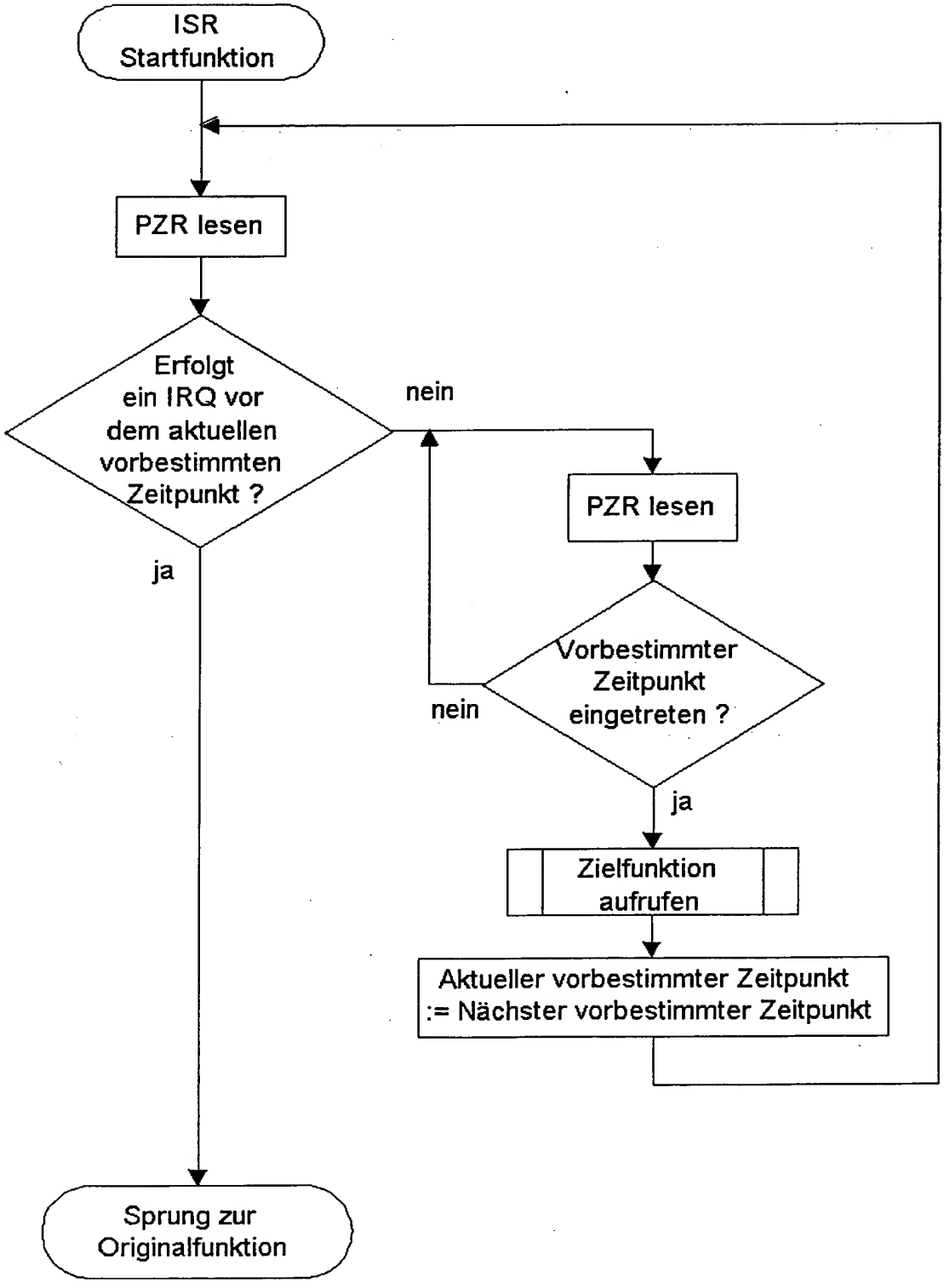

ISR Startfunktion
PZR lesen
Aktuellen Zeitverzug bestimmen
Aktueller Zeitverzug größer als zeitlicher Vorlauf ?
nein
ja
Zeitlicher Vorlauf := Aktueller Zeitverzug * Sicherheitsfaktor
PZR lesen
nein
Vorbestimmter Zeitpunkt eingetreten ?
ja
Polling
Timer auf nächsten vorbestimmten Zeitpunkt abzüglich des zeitlichen Vorlaufs programmieren
Zieltermin ?
nein
ja
Zielfunktion aufrufen
nein
Doppeltermin ?
ja
IRET
Sprung zur Originalfunktion

FIG. 4


ISR
Startfunktion
PZR lesen
Erfolgt
ein IRQ vor
dem aktuellen
vorbestimmten
Zeitpunkt ?
nein
ja
PZR lesen
Vorbestimmter
Zeitpunkt
eingetreten ?
nein
ja
Zielfunktion
aufrufen
Aktueller vorbestimmter Zeitpunkt
:= Nächster vorbestimmter Zeitpunkt
Sprung zur
Originalfunktion